Towards Twin-Driven Engineering: Overview of the State-of-the-Art and Research Directions

Massimo Tisi, Hugo Bruneliere, Juan de Lara, Davide Di Ruscio, Dimitris Kolovos

Towards Twin-Driven Engineering: Overview of the State-of-the-Art and Research Directions

Massimo Tisi[1], Hugo Bruneliere[1], Juan de Lara[2], Davide Di Ruscio[3] and Dimitris Kolovos[4]

[1] IMT Atlantique, LS2N (UMR CNRS 6004), Nantes, France
{massimo.tisi,hugo.bruneliere}@imt-atlantique.fr
[2] Universidad Autónoma de Madrid, Madrid, Spain
juan.delara@uam.es
[3] University of L'Aquila, L'Aquila, Italy
davide.diruscio@univaq.it
[4] University of York, York, United Kingdom
dimitris.kolovos@york.ac.uk

Abstract. Cyber-Physical Systems (CPS) are complex physical systems interacting with a considerable number of distributed computing elements for monitoring, control and management. They are currently becoming larger as Cyber-Physical Systems of Systems (CPSoS), since many industrial companies are transitioning their complex systems of systems to software-intensive solutions in different domains such as production or manufacturing. Following the development and dissemination of DevOps approaches in the Software Engineering world, we propose the Twin-Driven Engineering (TDE) paradigm as a way to upgrade the role of Digital Twins (DT) to become a central point in all the engineering activities on the CPSoS, from design to decommissioning. Since CPSoS can be highly heterogeneous, we rather target the support for producing and maintaining a single integrated virtual representation of the CPSoS (i.e. a System of Twins) on which it is possible to perform global reasoning, analysis and verification. However, such a new paradigm comes with several open research challenges. We provide an overview of the state-of-the-art in key areas related to TDE. We identify under-investigated problems in related work and outline corresponding research directions.

Keywords: Twin-Driven Engineering, Cyber-Physical Systems, Systems of Systems, State-of-the-Art, Research Directions.

1 Introduction and Motivation

Cyber-Physical Systems (CPS) are complex physical systems interacting with a large number of distributed computing elements for monitoring, control and management. Cyber-Physical Systems of Systems (CPSoS) further leverage connectivity to achieve complex tasks by "*an integration of a finite number of constituent systems which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal*" [15]. In this context, Digital Twins (DTs), i.e. virtual

real-time representations of physical systems, are particularly relevant for monitoring and making diagnostics/prognostics on the constituent CPSs. For example, virtual representation has been already used for virtual commissioning of manufacturing systems [21]. However, the real-time connection of digital twins with the physical system while it is in use enables further usages, like seamless tracking of the system operation and more realistic evaluation of any system improvements. More generally, in current practices and different domains, the development and maintenance of a CPSoS is performed through the interaction of multidisciplinary actors with specific competences and assigned to different phases. To capture, communicate and validate their ideas with other stakeholders, engineers of each CPSoS constituent system build models of their respective systems. However, the traditional paradigm "*requirements - model-based design - verification - deployment - operation - decommissioning*" does not provide sufficient guarantees to the dependability of a CPSoS. Indeed, miscommunication and delays in propagating changes are a primary cause of faults for CPSoS. In Software Engineering, the DevOps approach is widely used to address similar problems by providing practices, techniques and tools for integrating the software development and operation phases. Interestingly, the application of DevOps approaches has been recently studied in the context of CPSs [8] [26]. We advocate that DevOps practices may have a strongly beneficial impact on the dependability of CPSoSs.

In this paper we propose to go further and to create a new engineering paradigm, Twin-Driven Engineering (TDE), with the aim of upgrading the role of DTs as a central point of all the engineering activities on the CPSoS. Since CPSoS heterogeneity is inevitable, our proposition is to help in producing and maintaining a single integrated virtual representation of the CPSoS on which it is possible to perform global reasoning, analysis and verification. Such representation of the CPSoS will orchestrate the DTs of the constituent systems: We call it a System of Twins (SoT). Such an approach can radically change the way CPSoS are produced and maintained. A SoT can precede the design of a constituent system and drive its actual development, or act as a specification and automated oracle for continuous engineering. The industrial deployment of this approach depends on the cost effectiveness of the SoT production and maintenance.

While inexpensive and easily interfaced sensors are available today, engineering a DT is still an expensive activity whose costs are impacted by their dependability requirements that span from certifying the conformance of the DT with the CPS, to robustness and runtime problems. Moreover, the required competences in Machine Learning (ML), e.g. for inferring DT non-measurable properties, are not generally available in the ecosystem. Finally, providing a global view (even abstract) of the full CPSoS may introduce new scalability requirements on the underlying infrastructure.
Hence, there is the need for research methods, techniques and tools to minimize the cost of the DT production at all levels of a multi-layered CPSoS. In this paper, we define TDE and provide an overview of the state-of-the-art in key related areas. We also introduce corresponding research directions we believe to be worth investigating.

Paper organization. Section 2 introduces the TDE paradigm. Section 3 presents the current state-of-the-art related to key challenges associated with the realization of TDE. Finally, Section 4 concludes the paper.

2 Twin-Driven Engineering

TDE can be defined as the software & system engineering paradigm that consists in creating, maintaining and leveraging a dependable twin of a given complex system and its environment, in order to better support and manage this system throughout its whole life cycle. As a consequence, in order to build the twin of a CPSoS, a corresponding SoT actually has to be specified and handled.

In this context, TDE first requires the users to describe the CPSoS including *i)* the structure of the system of systems, *ii)* the interaction of the constituent systems and *iii)* the structure and behavior of the environment. Information on each constituent system is also needed, without delving into full implementation details at that level. We refer to this general information as *Abstract Models*, i.e. single sources of truth on their constituent systems as shown in Figure 1. To be able to specify these Abstract Models, the user needs a modeling language. Each domain in CPSs has its own set of commonly used description languages and models, referring to different physical, engineering and technological backgrounds. Models may have different fidelity, different types of parameterization, or different philosophical approaches. Some academic literature and related work have tried to propose very general languages for describing any CPS (cf. Section 3). However, the technical and organizational heterogeneity of CPSoS strongly reduces the realistic chances of acceptance of such efforts.

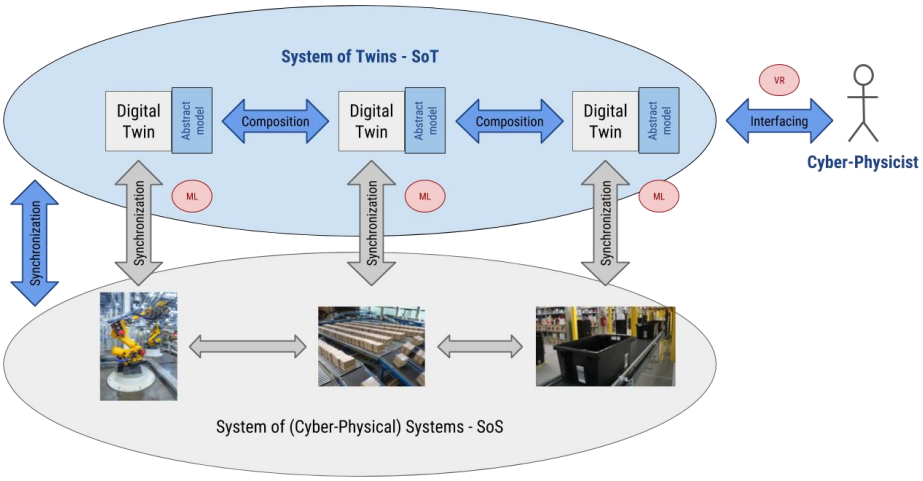

Fig. 1. An overview of the Twin-Driven Engineering paradigm, showing the central role of the Digital Twins (DTs) that compose the System of Twins (SoT) to be synchronized with the underlying System of Systems (SoS).

As a solution, we propose to consider a specific DT for each constituent system. The composition of all these DTs will result in the target SoT that can then be used as the main interface between the underlying CPSoS and the involved engineers. To make TDE possible, the SoT has to be kept constantly synchronized with the related

CPSoS. Thus, each DT has to be individually synchronized with its underlying system. At this stage, we envision the use of ML techniques to ensure such a regular synchronization. Moreover, each DT is also in charge of keeping its corresponding Abstract Model up to date. Solutions already exist to simplify the creation of DTs (cf. Section 3). However, they generally rely on stakeholders committing to a single technology, e.g. a specific language, development tool, runtime infrastructure, or dependability definition. This is unrealistic if we consider the growing organizational and technical complexity of CPSoS. Thus, the key idea coming along with TDE is to propose a corresponding meta-environment for building specific DTs tailored for each CPSoS. By using such a platform, engineers operating at different levels of the CPSoS hierarchy can create solutions to produce an integrated view of their component systems for global reasoning. As mentioned before, TDE does not aim at replacing existing tools and techniques for modeling individual components of CPSoS. The objective is rather to build on top of them, e.g. by using them as targets of code generation and directly integrating their executable artifacts.

3 Advancing the State-of-the-Art of Twin-Driven Engineering

3.1 Definition of domain-specific Systems of Twins

Current metamodeling languages (e.g. MOF [25], Ecore [7]) were devised for static data modeling. However, describing SoTs and coupling them with design models of CPSs requires *i)* accounting for dynamic data and *ii)* the availability of CPS metamodeling primitives to describe quantity units, time models, probability and uncertainty. Thus, using current metamodeling frameworks would require extra effort, because CPS-specific concepts need to be explicitly modeled. Instead, a native support would reduce the effort needed to build domain-specific CPS languages tools. These primitives need to be properly integrated with languages for model manipulation or constraints (e.g. OCL). As a consequence, TDE requires pushing forward the state-of-the-art on metamodeling technologies for engineering SoTs.

Uncertainty and quantity units have already been introduced in UML/OCL [23]. Later, some UML/OCL types were also extended with uncertainty [3]. While TDE could leverage on those works, these are not realized in standard metamodeling frameworks and so not available for building practical DSLs. Ecore has been recently extended with temporal capabilities [12]: Updates of temporal elements in a model are persisted and it is possible to issue temporal queries, or to retrieve elements in previous model versions. These ideas can be used as a basis to extend with proper time primitives like explicit clock-times or intervals. More sophisticated notions of time have been added to OCL [17], enabling quantification of events and over time. These works can be extended with stream types for modeling live data, to check the runtime behavior of SoT.

Many works have been proposed on specification-based monitoring of CPS [2], most of them based on variations of temporal logic. Hence, there is a frequent need to tweak existing logic formalisms and create supporting tools, which is a very costly activity. TDE aims at providing means to define DSLs for runtime monitoring, and the support for the automated creation of monitors. TDE will also provide support for

linking design-time models (e.g. in Modelica), and TD models by means of bidirectional (bx) domain-specific transformation languages [9]. Bx will enable iterative design processes, while our vision is to provide a framework for engineering bx languages targeting specific CPS design platforms like Modelica. Many works exist on bx transformation languages [14]. However, we are not aware of frameworks for engineering domain-specific bx transformation languages. Additionally, the use of model view techniques could also be considered [5] as they have already demonstrated their relevance and applicability in the context of the federation of large-scale design time and runtime models [6].

Research directions. TDE takes a step beyond the state-of-the-art frameworks for defining SoT by providing advanced metamodeling capabilities, and bidirectional (BX) connection with design CPS models, for a holistic support of the CPS life cycle. Standard metamodeling frameworks have to be extended with CPS notions (quantities, time, uncertainty) as well as types for dynamic and streaming data (enabling e.g., connection with predictive models). Standard constraint languages also have to be extended to enable expressing time-aware constraints. Such constraints can natively use CPS concepts and be used as a basis to develop monitoring languages for dependability properties. Finally, domain-specific BX transformation languages have to be specified to facilitate the creation of bridges between SoT and CPS design models. Such languages also have to support the checking of CPS correctness properties, involving the bx preservation of source and target integrity constraints.

3.2 Domain-specific 3D interfaces for Systems of Twins

Arguably, a domain-specific language for SoTs is of little value if it is not supported by robust and usable editors through which developers can create and edit models conforming to the language. In the field of 2D graphical editor development, frameworks such as MetaEdit+, GMF, Graphiti and Sirius have drastically reduced the effort and expertise required to develop and maintain diagram-based editors for domain-specific languages [18]. Conversely, there is virtually no support for developing domain-specific 3D editors, except for some early prototypes that are no longer maintained [28]. 2D graphical editors may be sufficient for some classes of domain-specific SoTs languages. However, when physical objects are modelled, a 3D editor can provide a more natural and intuitive representation that better communicates to other engineers, and also to non-technical stakeholders. Developing such a 3D editor for a SoT DSL involves a significant upfront investment and requires specialized skills that would be prohibitive for most development teams. As discussed above, this used to be the case for 2D editors as well until tools such as MetaEdit+ and GMF provided reusable facilities that allowed engineers to concentrate only on the essential complexity of the editor (e.g. define how abstract syntax concepts should be mapped to shapes/connections).

Research Directions. TDE requires a set of notations for specifying the 3D graphical syntax of SoT domain-specific languages at a high level of abstraction, and automated facilities for realizing fully-functional 3D editors for related models. It also requires exploring virtual reality (VR) technologies through which users can be able to navigate and interact with such 3D models, as well as protocols for capturing the structural/behavioral properties of the language in a platform-independent way. A

main objective is to facilitate concrete implementations on different platforms, similarly to Microsoft's Language Server Protocol (LSP) for textual languages and GLSP for graphical languages.

3.3 Dependable connection and composition in Systems of Twins

Solutions to simplify the development of DTs like ADT, Eclipse Ditto, or Seebo define or generate an API for the DT on a specific platform (e.g. Azure IoT), with a specific software architectural style (e.g. REST on Swagger) and/or with an organization that is agnostic of the CPSoS behavior. A mechanism is needed to generate, for any given platform, protocol and style, an efficient interface for the DT.
An important dependability issue, which has not yet been addressed in production systems such as ADT, is related to the way the CPSoS are connected to the DT [20]. CPSoS are not managed by actors such as Microsoft, they belong to third party players and are connected to the cloud platform through the internet backbone (generally WAN links operated by another actor). Hence, it is critical to consider the CPSoS, the DT and the infrastructure overall. Similarly, to the studies that have been conducted for smart-grid applications, it is critical to identify the requirements of each DT and the possible limitations of the infrastructure in order to ensure a DT can collect the metrics it needs to maintain the DT up-to-date. Going further, strategies to cope with network disconnections should be defined à priori.

Monitoring CPSoS requires fast and low-overhead model analysis and transformation techniques, to ensure that corrective decisions are timely and achieved with acceptable use of resources (CPU, memory, bandwidth and energy). While such techniques are not yet available for CPS, relevant advances have been made for software systems through the development of incremental techniques for the efficient analysis of functional requirements expressed in OCL [10] and of non-functional requirements specified in probabilistic temporal logics [16]. However, because of the high throughput of incoming data, a dependable continuous monitoring of DT models requires techniques that exceed the typical domain of incremental computation, to enter the area of streaming computation.

Research Directions. A DT solution should come with the right mechanism/components [24] to deal with all infrastructure specifics (throughput, latency and resilience of the network) so that we can formally validate whether the DT can be correctly maintained in time. Those mechanisms can include compression, caching and resynchronization mechanisms according to the CPSoS specifics, the DT expectations and the infrastructure. By leveraging previous works on software programming models [11] and middleware for IoT [27], TDE can rely on a DSL that enables DevOps to express the infrastructure topology, its specific requirements in terms of connectivity between each CPSoS and the SoT. A list of components in charge of dealing with interconnectivity issues and network specifics (latency/throughput) can be defined and then used to derive a DSL allowing to express those constraints. The ultimate goal would be to automatically generate, configure and finally deploy those components.

3.4 Twin-Driven Engineering for DevOps on CPSoS

Managing CPS development is a complex task because different kinds of artifacts (from physical to software systems) need to be homogeneously integrated. In this context, DTs come to the rescue by playing the role of bridges between the physical and digital worlds. They allow identifying problems even before they occur, or enabling analysis and simulations to enhance system dependability. However, in CPSoSs, a multitude of heterogeneous DTs must be managed and properly interconnected. Moreover, the same physical object can have different DTs providing different abstractions and only exposing the characteristics appropriate for the considered life cycle phase. To enable TDE, two main ingredients are needed: *1)* A dedicated repository able to store and manage reusable DT models and *2)* a DevOps infrastructure supporting the SoT continuous engineering and integration.

Concerning *(1)* we outline research challenges for achieving a comprehensive solution to the problem of properly managing the persistence of models and the discovery of any kind of modeling artefact and tool for enabling their reuse and refinement. CDO is a pure Java model repository (relying on common database backends) for models that can also serve as a persistence and distribution framework for model-based applications. EMFStore [19] is a model version control system implementing the typical operations proposed by SVN/CVS/Git for text-based artefacts. GME - Generic Modeling Environment [22] is a set of tools supporting the creation of domain specific modeling languages and code generation environments, based on MS repository technologies to store the developed models. ModelBus [13] consists of a central bus-like communication infrastructure, a number of core services (versioning, check-out, merging) and a set of additional management tools. MDEForge [1] is an extensible Web-based modeling platform fostering community-based modeling repositories and proposing remote model management tools as software-as-a-service.

Concerning *(2)* several tools support the continuous integration and deployment of software systems (e.g., Jenkins, Travis, and GitLab CI/CD). “Continuous Integration” was first used by Grady Booch [4] to describe an effective, iterative way of building software. Essentially, every commit immediately triggers automated tests and building tasks to quickly detect errors and constantly have a stable build of the system being developed. In complex CPSoS, the existence of large numbers of decentralized sub-systems represents an additional element of complexity. Finally, CPSoS are also characterized by human-in-the-loop aspects related to the collaboration of humans with the software and hardware systems being employed.

Research Directions. TDE requires to develop software components, available as software-as-a-service, to support the persistence and the reuse of DT models: Tools from different vendors (e.g., Matlab Simulink, LabVIEW, Modelica) could store and retrieve models from a common repository. Moreover, a dedicated support has to be implemented to manage the relationships between heterogeneous artefacts, including advanced query mechanisms. TDE also comes with novel methods and tools for supporting SoT development and operations. Dependability models and processes have to be investigated to guarantee reliability during the complete life cycle of CPSoS, from requirements capture to design, test, operation and decommissioning. It can be explored how existing DevOps infrastructures can be extended to support the continuous engineering and continuous integration of SoTs.

4 Conclusion

In this paper, we proposed a first definition of the TDE paradigm for CPSoS. We gave an overview of the state-of-the-art in key related areas, and proposed corresponding research directions we think to be worth investigating. We believe cost-effective TDE to be a major milestone for dependable software engineering for production systems. Thus, this paper also aims at fostering discussion and collaboration around this topic between the software engineering and production systems communities.

References

1. F. Basciani, J. Di Rocco, D. Di Ruscio, A. Di Salle, L. Iovino, et A. Pierantonio, MDEForge: an extensible Web-based modeling platform, CloudMDE Workshop at MoDELS 2014, Valencia, Spain (2014).
2. E. Bartocci et al. Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications, Lectures on Runtime Verification - Introductory and Advanced Topics, vol. 10457, Springer, p. 135–175 (2018).
3. M. F. Bertoa, N. Moreno, G. Barquero, L. Burgueño, J. Troya, A. Vallecillo. Expressing Measurement Uncertainty in OCL/UML Datatypes, ECMFA 2018, vol. 10890, p. 46–62 (2018).
4. G. Booch. Object-Oriented Design with Applications, The Benjamin/Cummings Publishing Company, Inc., (1991).
5. H. Bruneliere, E. Burger, J. Cabot and M. Wimmer. A Feature-based Survey of Model View Approaches. Software & Systems Modeling, 18(3), pp.1931-1952 (2019).
6. H. Bruneliere, F. Marchand de Kerchove, G. Daniel, S. Madani, D. Kolovos and J. Cabot. Scalable Model Views Over Heterogeneous Modeling Technologies and Resources. Software and Systems Modeling, 19(4), pp.827-851 (2020).
7. F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, et T. J. Grose. Eclipse Modeling Framework. Addison Wesley (2003).
8. B. Combemale et M. Wimmer. Towards a Model-based DevOps for Cyber-Physical Systems. In International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment (pp. 84-94). Springer, Cham, (2020).
9. J. S. Cuadrado, E. Guerra, et J. de Lara. Towards the Systematic Construction of Domain-Specific Transformation Languages, ECMFA 2014, vol. 8569, p. 196–212 (2014).
10. A. Demuth, R. E. Lopez-Herrejon, et A. Egyed. Automatic and Incremental Product Optimization for Software Product Lines, ICST 2014, p. 31–40 (2014).
11. I. Gerostathopoulos et al. Self-adaptation in Software-intensive Cyber–Physical Systems: From System Goals to Architecture Configurations, Journal of Systems and Software, vol. 122, p. 378–397 (2016).
12. A. Gómez, J. Cabot, et M. Wimmer, TemporalEMF: A Temporal Metamodeling Framework, ER 2018, vol. 11157, p. 365–381 (2018).
13. C. Hein, T. Ritter, et M. Wagner. Model-driven Tool Integration with ModelBus, Workshop on Future Trends of Model-Driven Development (2009).
14. S. Hidaka, M. Tisi, J. Cabot, et Z. Hu. Feature-based Classification of Bidirectional Transformation Approaches, Software and Systems Modeling, vol. 15, no 3, p. 907–928 (2016).

15. M. Jamshidi, Systems of Systems Engineering: Principles and Applications. CRC press, (2008).
16. K. Johnson, R. Calinescu, et S. Kikuchi. An Incremental Verification Framework for Component-based Software Systems, CBSE 2013, p. 33–42 (2013).
17. B. Kanso et S. Taha. Specification of Temporal Properties with OCL, Science of Computer Programming, vol. 96, p. 527–551 (2014).
18. D. S. Kolovos, A. García-Domínguez, L. M. Rose, et R. F. Paige. Eugenia: Towards Disciplined and Automated Development of GMF-based Graphical Model Editors, Software and Systems Modeling, p. 1–27 (2015).
19. M. Koegel et J. Helming. EMFStore: A Model Repository for EMF Models, ICSE 2010, vol. 2, p. 307-308 (2010).
20. E. A. Lee. Cyber Physical Systems: Design Challenges, ISORC 2008, p. 363–369 (2008).
21. C.G. Lee et S.C. Park. Survey on the Virtual Commissioning of Manufacturing Systems. Journal of Computational Design and Engineering, 1(3), pp.213-222 (2014).
22. A. Ledeczi et al. The Generic Modeling Environment, Workshop on Intelligent Signal Processing (2001).
23. T. Mayerhofer, M. Wimmer, et A. Vallecillo. Adding Uncertainty and Units to Quantity Types in Software Models », MODELS 2016, p. 118–131 (2016).
24. M. Mikic-Rakic et N. Medvidovic. A Classification of Disconnected Operation Techniques, EUROMICRO 2006, p. 144–151 (2006).
25. The Object Management Group, OMG's Meta-Object Facility (MOF), https://www.omg.org/mof/, last accessed 2021/06/18.
26. M.U. Querejeta, , L. Etxeberria et G. Sagardui. Towards a DevOps Approach in Cyber Physical Production Systems Using Digital Twins. International Conference on Computer Safety, Reliability, and Security, pp. 205-216. Springer, Cham, (2020).
27. M. A. Razzaque, M. Milojevic-Jevric, A. Palade, et S. Clarke, Middleware for Internet of Things: A Survey, IEEE Internet of Things Journal, vol. 3, no 1, p. 70–95, (2016).
28. J. Wolter et U. Kastens. Generating 3D Visual Language Editors: Encapsulating Interaction Techniques in Visual Patterns, International Journal of Software Engineering and Knowledge Engineering, vol. 25, no 2, p. 333–360, (2015).